Carnegie Mellon University
Heinz College

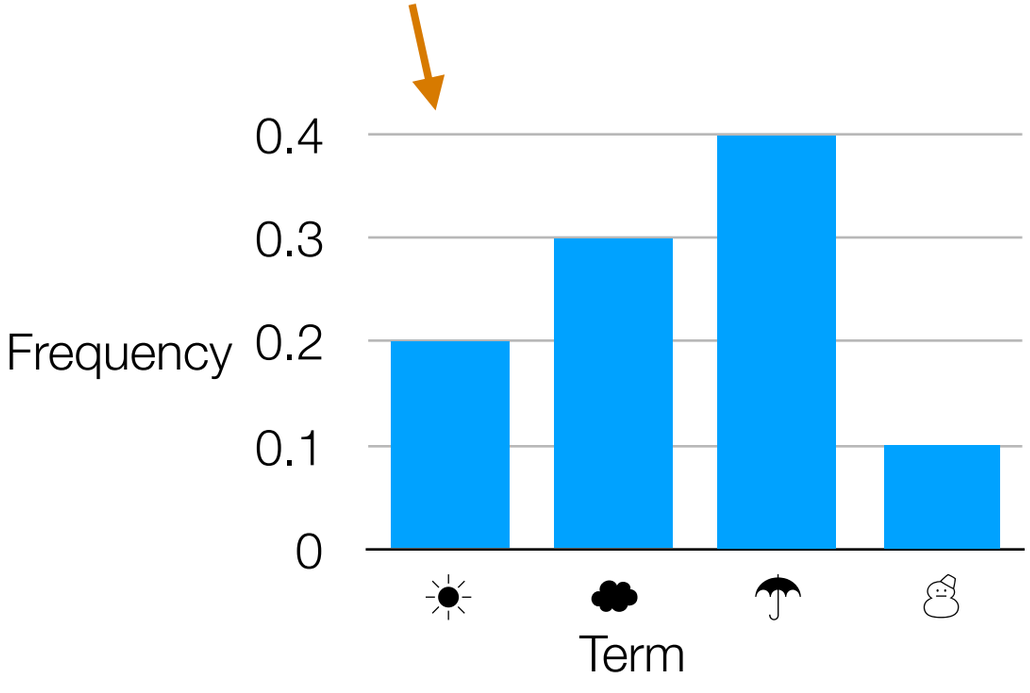
94-775 Lecture 6: Visualizing High-Dimensional Feature Vectors and Intro to Clustering

George Chen

Recap: Basic Text Analysis

- Represent text in terms of “features”
  (such as how often each word/phrase appears)
  - Can repeat this for different documents:
    represent each document as a “feature vector”

"Sentence": ☀☂☁☁☁☂☃☂☂☀



$$\begin{bmatrix} 0.2 \\ 0.3 \\ 0.4 \\ 0.1 \end{bmatrix}$$

This is a point in 4-dimensional space, $\mathbb{R}^4$

dimensions = number of terms

In general (not just text): first represent data as feature vectors

Last Time

Find interesting relationships between 2 items at a time (e.g., Elon Musk & Tesla)

But how to compare which news articles are similar?

How do we plot articles so that we can compare lots of them?

Bad idea: stare at ***n*** histograms where ***n*** is very large

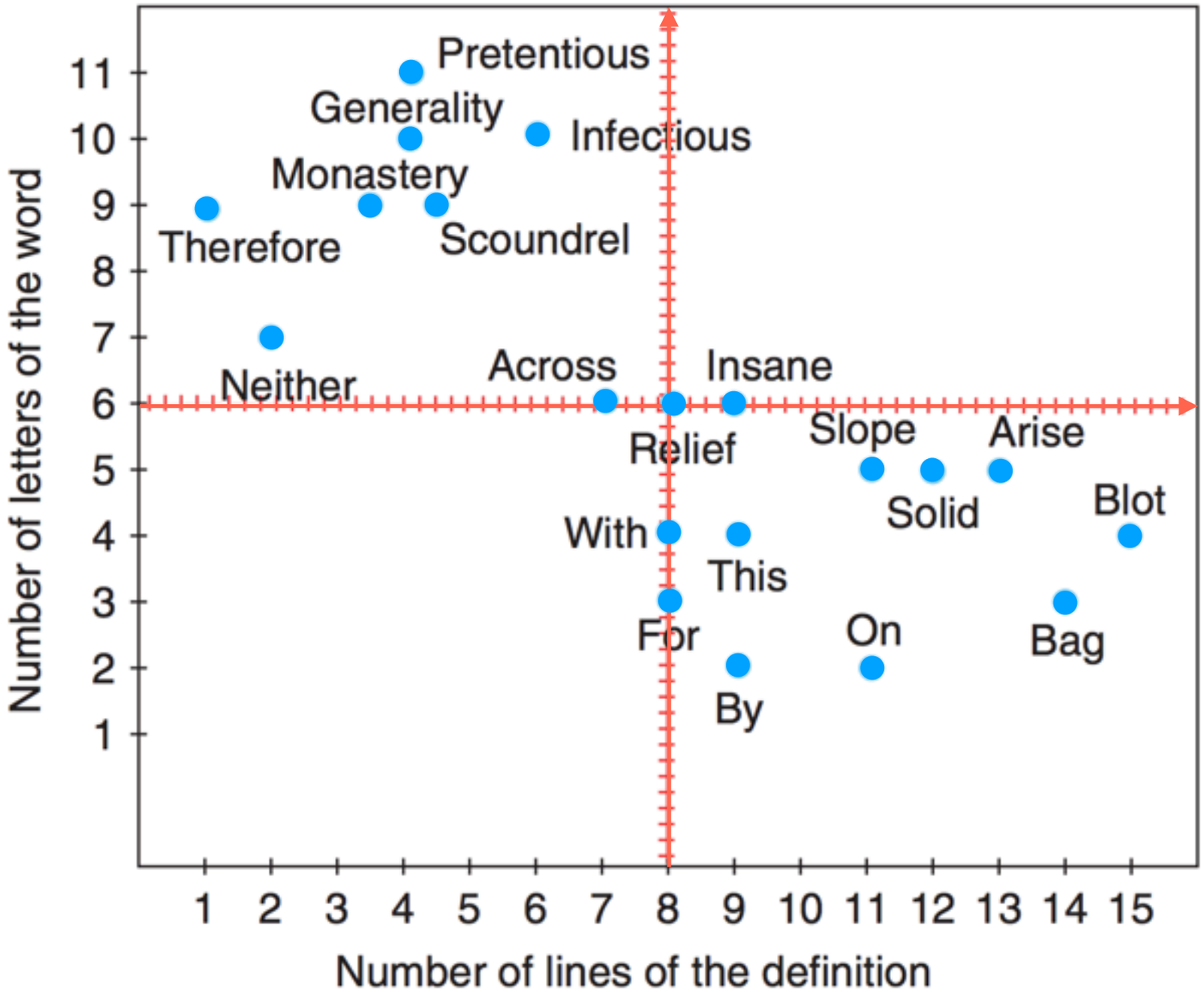
How do we identify whether there are groups of “similar” articles?

The issue is that as humans we can only really visualize up to 3 dimensions easily

Goal: Somehow reduce the dimensionality of the data preferably to 1, 2, or 3

Principal Component Analysis (PCA)

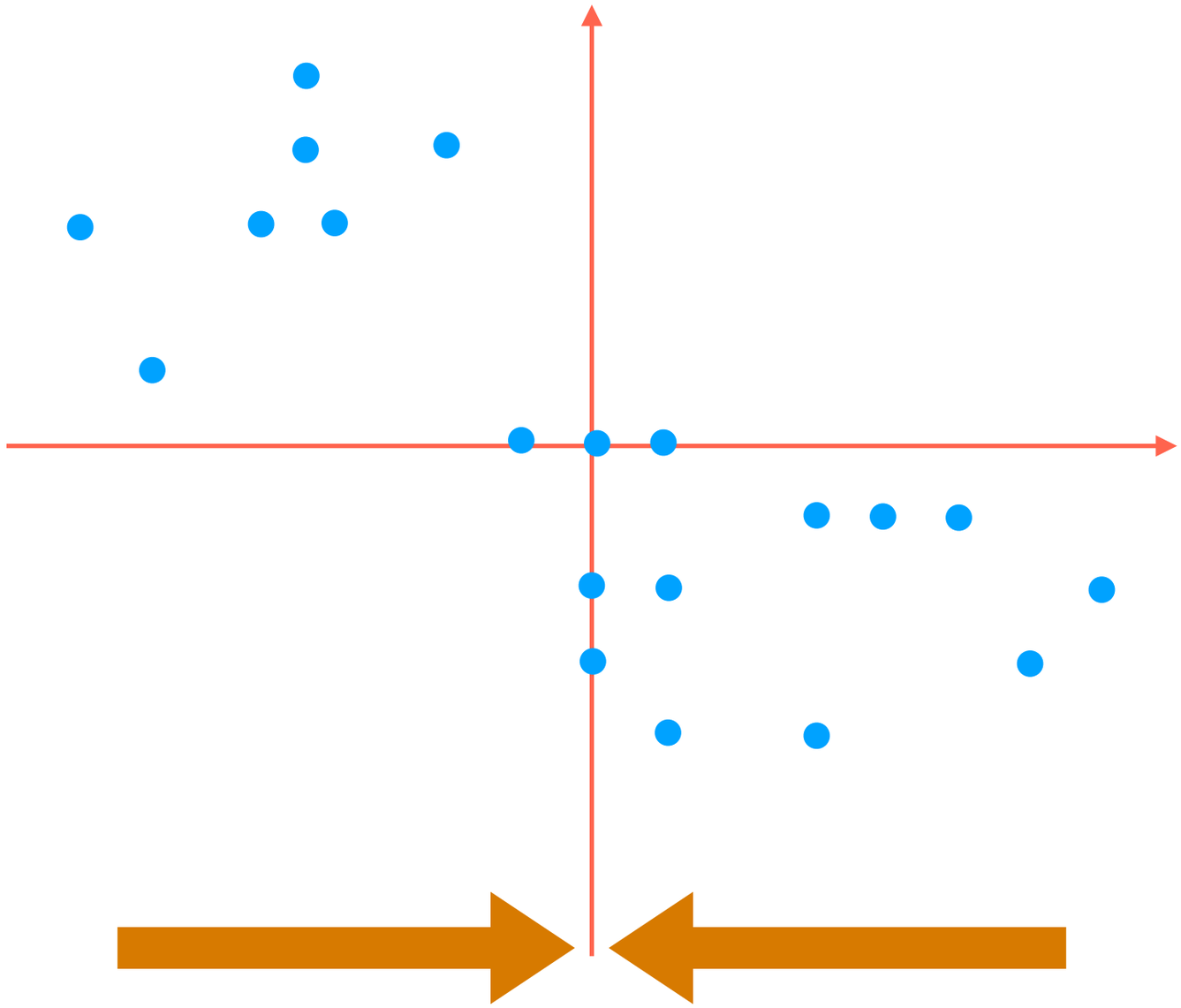
How to project 2D data down to 1D?



Hervé Abdi and Lynne J. Williams. Principal component analysis. Wiley Interdisciplinary Reviews: Computational Statistics. 2010.

Principal Component Analysis (PCA)

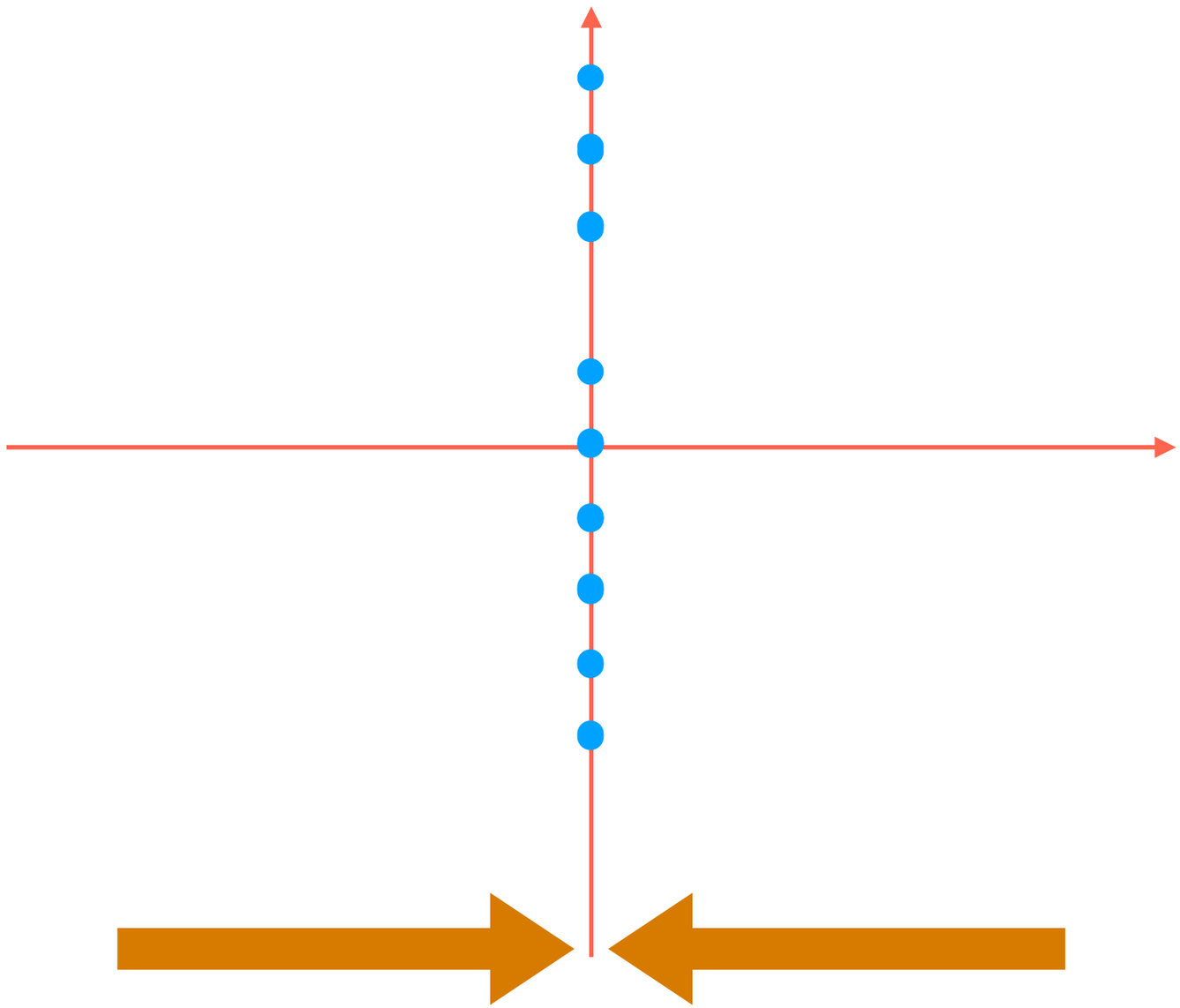
How to project 2D data down to 1D?



Simplest thing to try: flatten to one of the red axes

Principal Component Analysis (PCA)

How to project 2D data down to 1D?

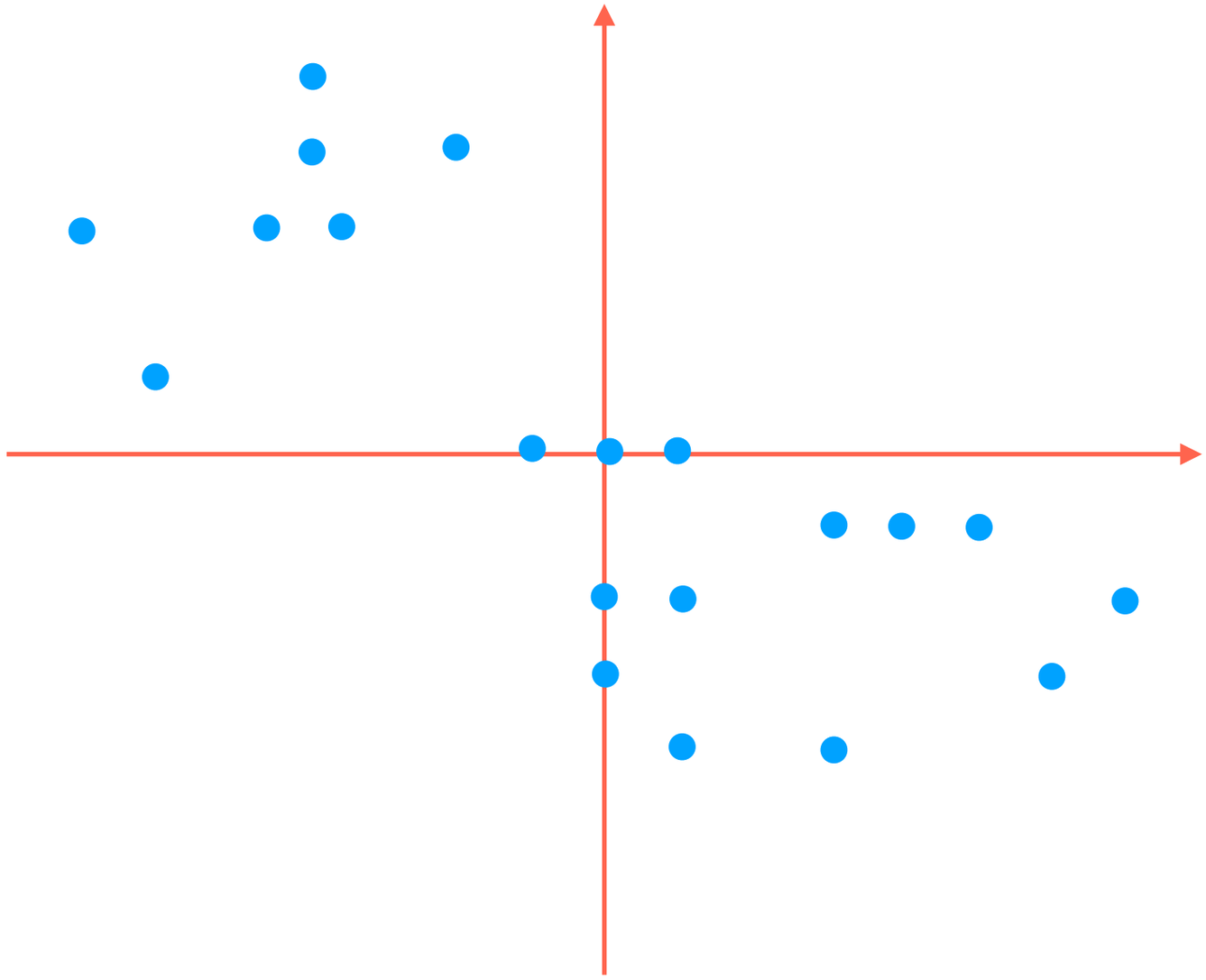


Simplest thing to try: flatten to one of the red axes

(We could of course flatten to the other red axis)

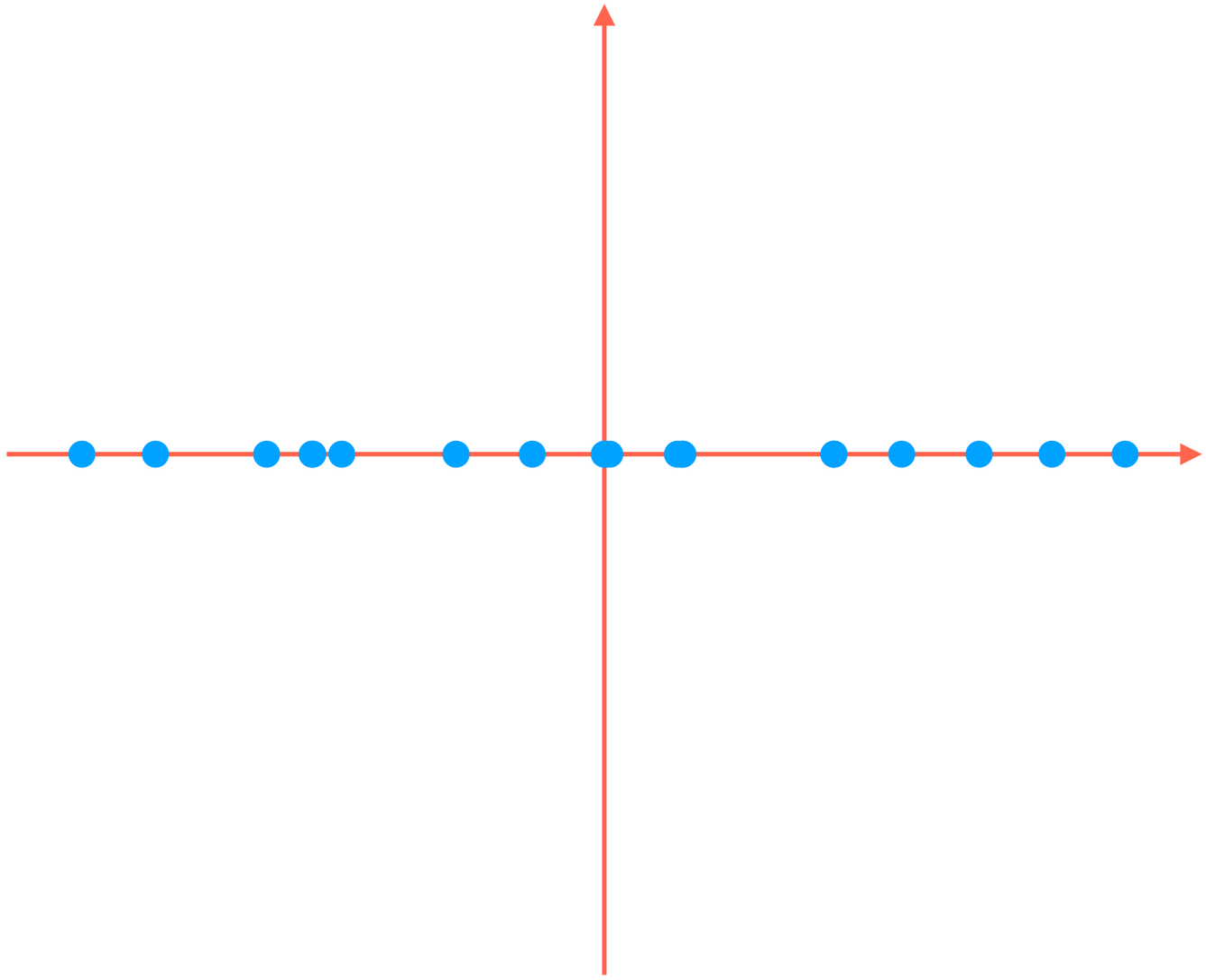
Principal Component Analysis (PCA)

How to project 2D data down to 1D?



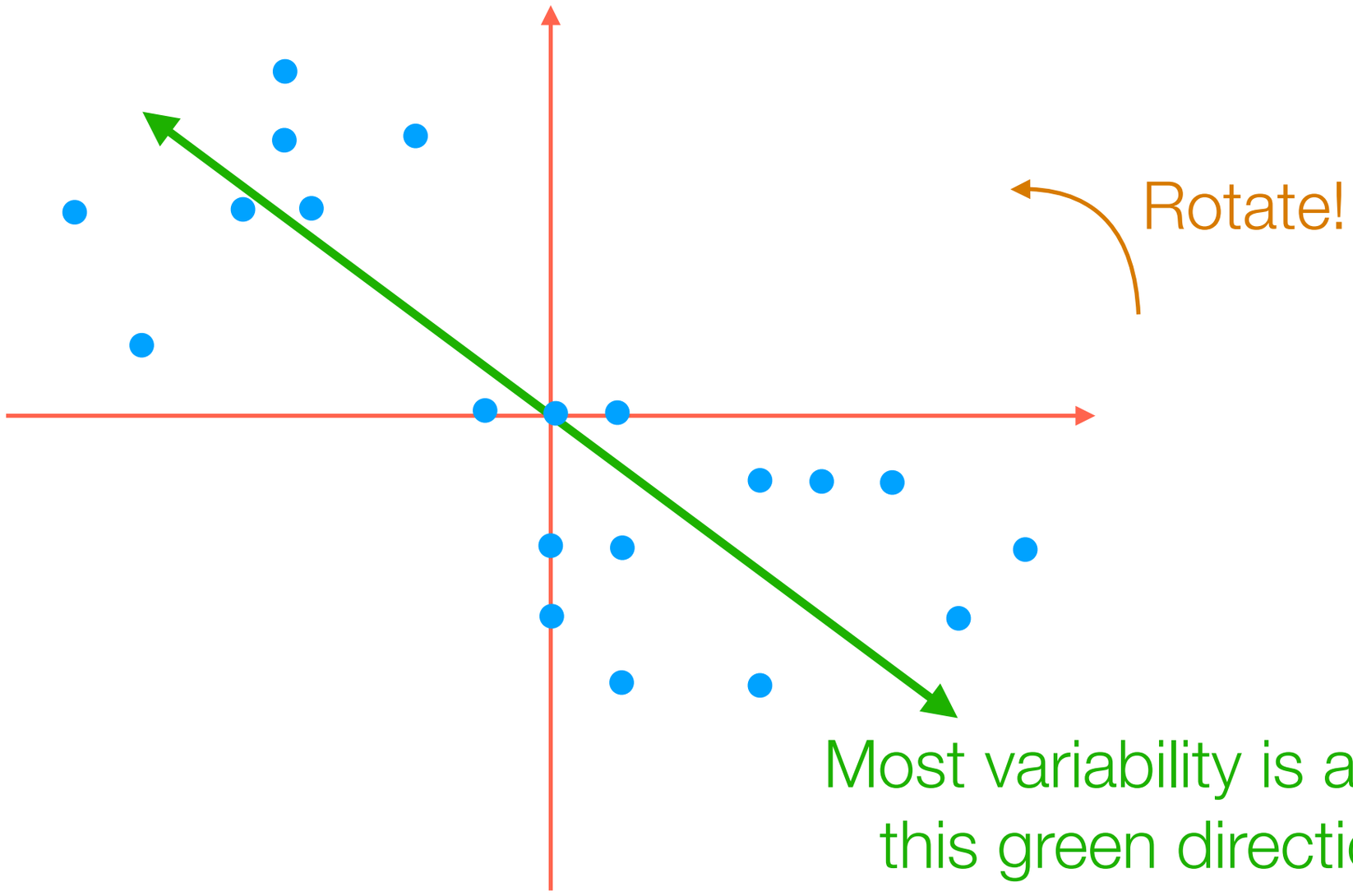
Principal Component Analysis (PCA)

How to project 2D data down to 1D?



Principal Component Analysis (PCA)

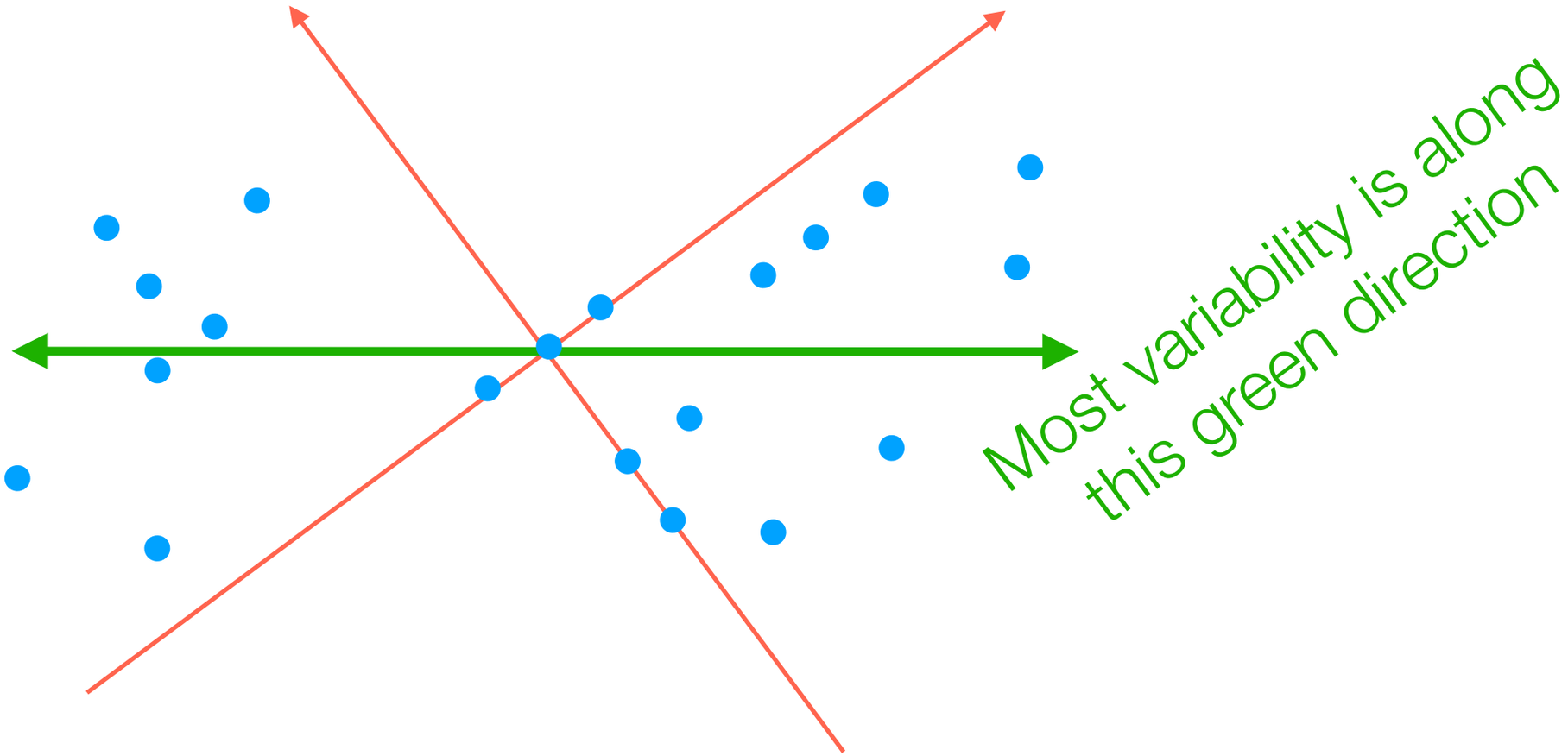
How to project 2D data down to 1D?



But notice that most of the variability in the data is *not* aligned with the red axes!

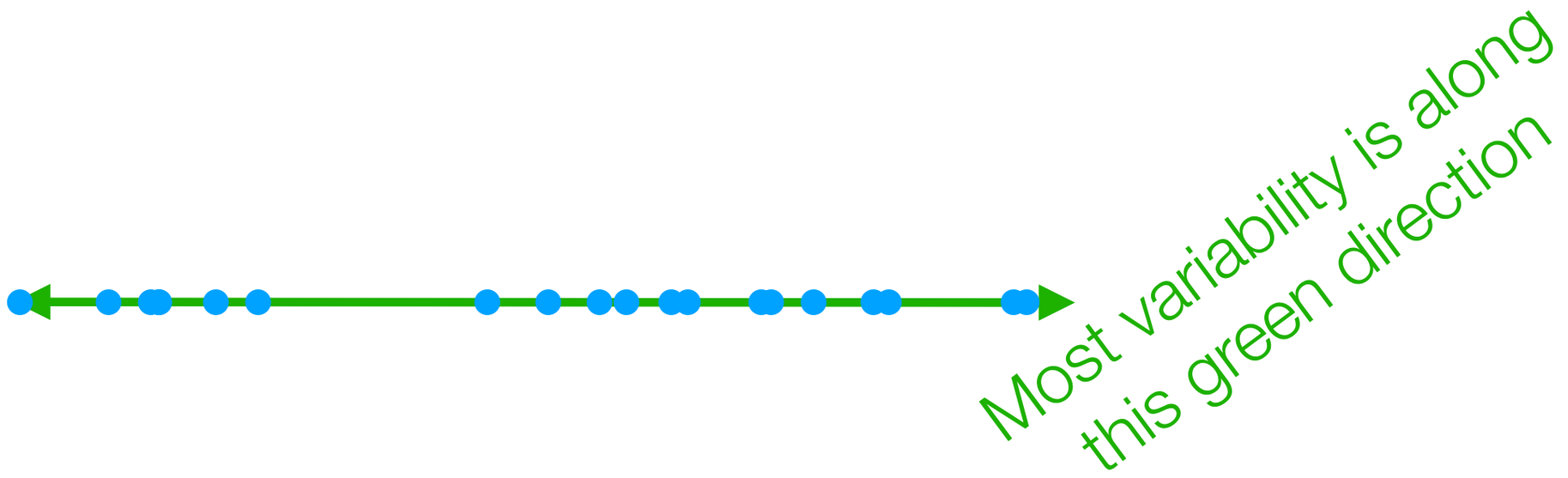
Principal Component Analysis (PCA)

How to project 2D data down to 1D?



Principal Component Analysis (PCA)

How to project 2D data down to 1D?



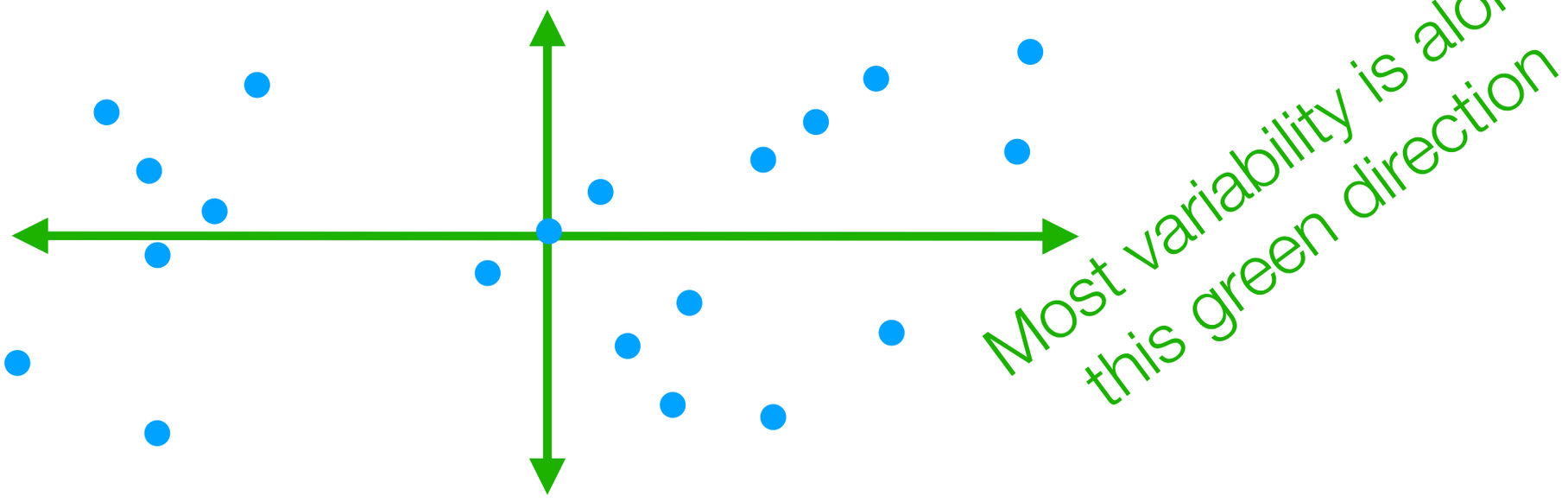
The idea of PCA actually works for 2D ➔ 2D as well
(and just involves rotating, and not “flattening” the data)

Principal Component Analysis (PCA)

~~How to project 2D data down to 1D?~~

How to rotate 2D data so 1st axis has most variance

before "flattening"



The idea of PCA actually works for 2D ➔ 2D as well
(and just involves rotating, and not "flattening" the data)

2nd green axis chosen to be 90° ("orthogonal") from first green axis

Principal Component Analysis (PCA)

- Finds top *k* orthogonal directions that explain the most variance in the data
  - 1st component: explains most variance along 1 dimension
  - 2nd component: explains most of remaining variance along next dimension that is orthogonal to 1st dimension
  - …
- "Flatten" data to the top *k* dimensions to get lower dimensional representation (if *k* < original dimension)

Dimensionality Reduction

- Large number of methods, not just PCA
- PCA is a linear method and is very well understood
- Many nonlinear dimensionality reduction methods often do better
  - Kind of like how nonlinear regression methods (e.g., kernel regression, Gaussian process regression, decision tree/forest regression) do better than linear regression
  - I will show results for one nonlinear dimensionality reduction method called t-SNE

Dimensionality Reduction

Demo

In many real datasets, clustering naturally arises!

Example: crime might happen more often in specific hot spots

Example: users in a recommendation system can share similar taste in products

Example: students have different skill levels
(clusters could correspond to different letter grades)

To come up with clusters, we first need to define what it means for two things to be “similar”

Defining Similarity

- There usually is no “best” way to define similarity

Example: cosine similarity $\dfrac{\langle Y_u, Y_v \rangle}{\|Y_u\| \|Y_v\|}$

- Also popular: define a distance first and then turn it into a similarity

Example: Euclidean distance $\|Y_u - Y_v\|$

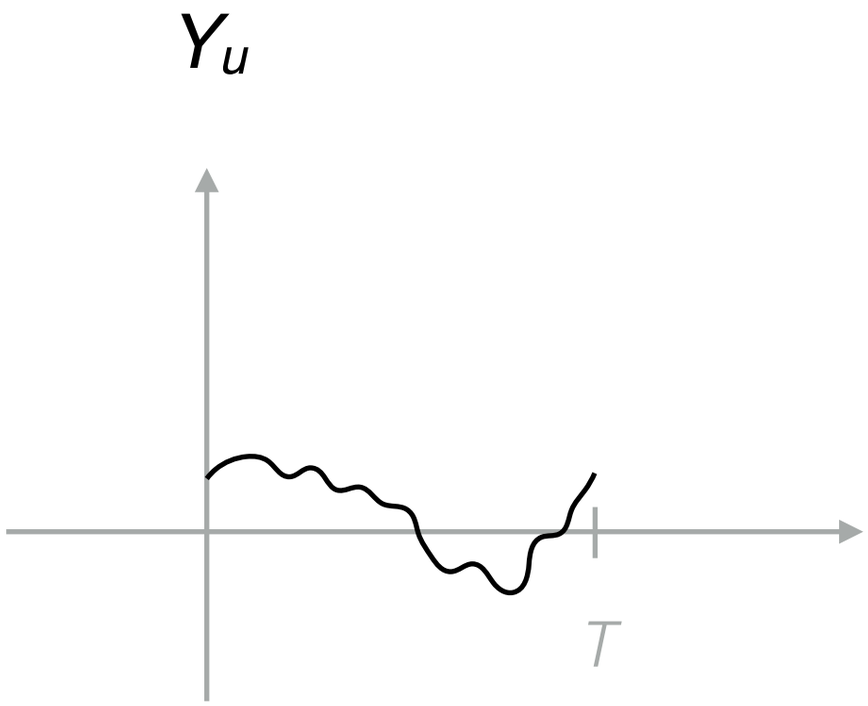
Turn into similarity with decaying exponential $\downarrow$

$$\exp(-\gamma \|Y_u - Y_v\|)$$

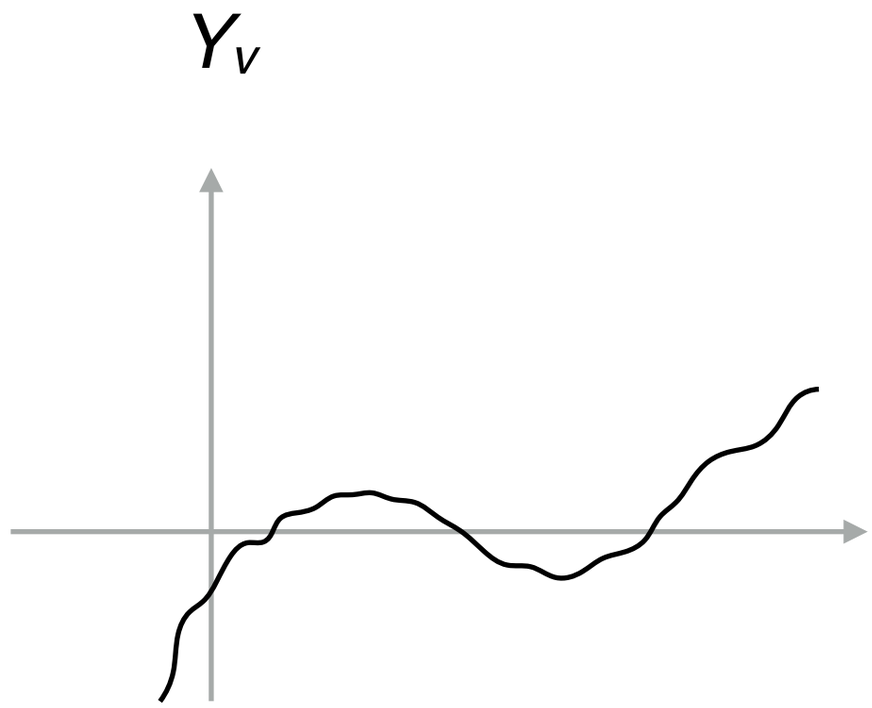
where $\gamma > 0$

Example: Time Series

How would you compute a distance between these?

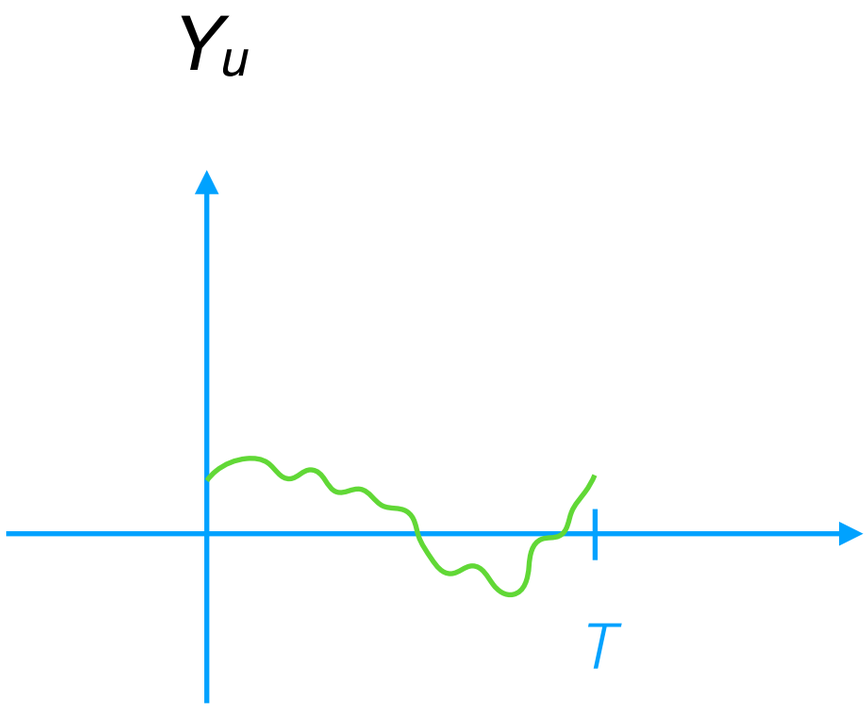


Only observe time steps between 0 and T

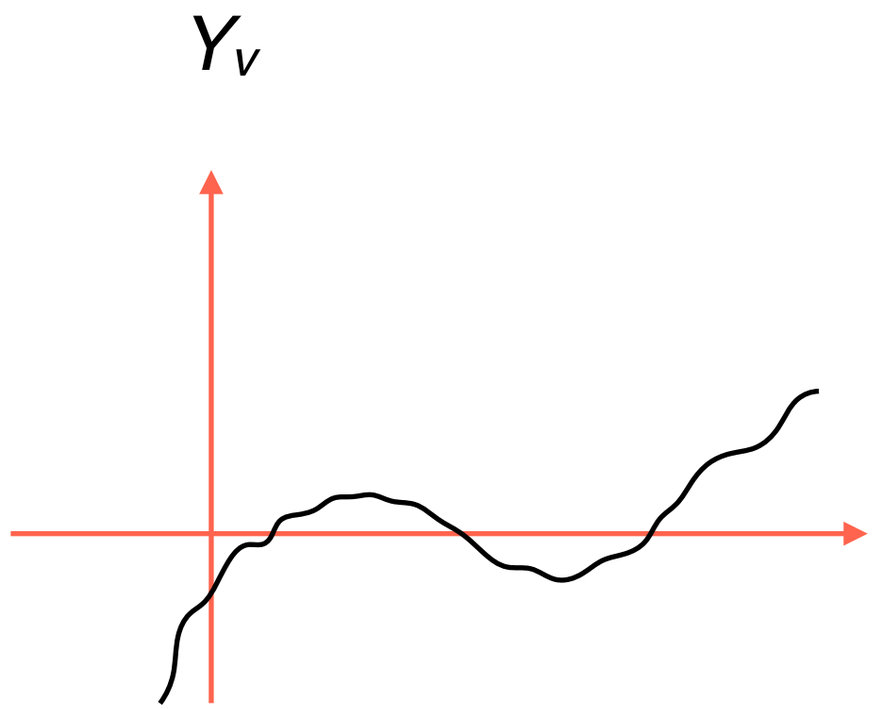


Example: Time Series

How would you compute a distance between these?

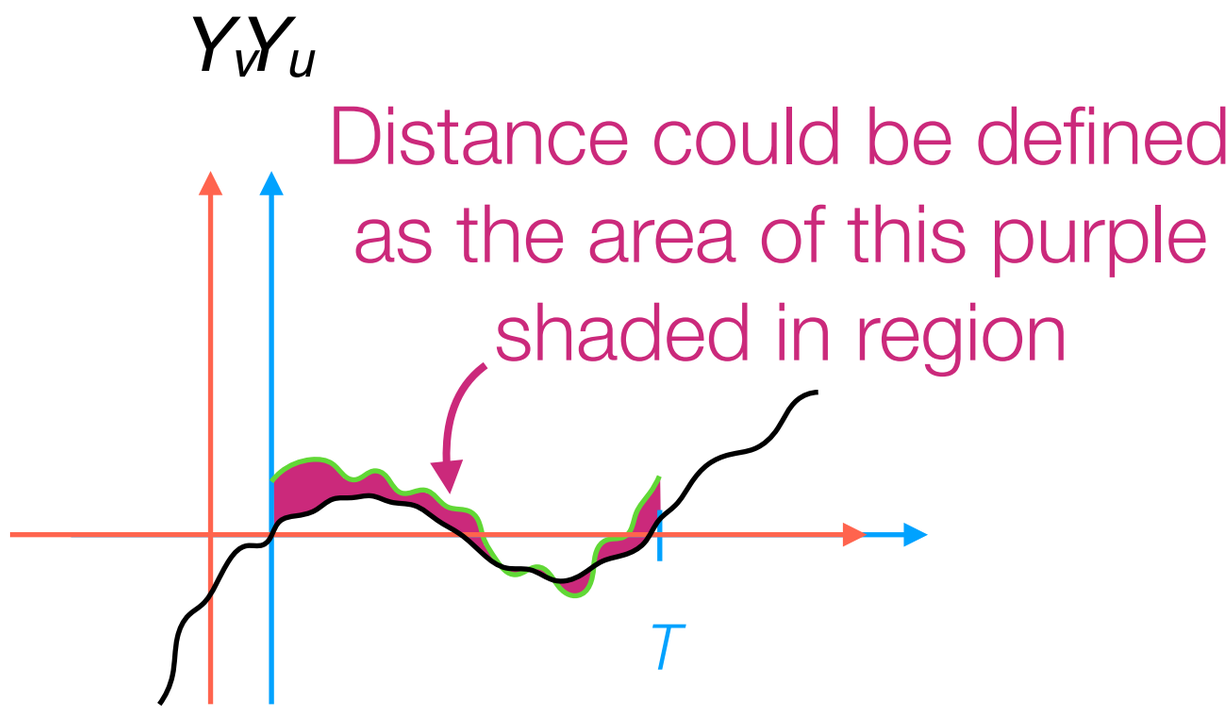


Only observe time steps between 0 and T



Example: Time Series

How would you compute a distance between these?



One solution: Align them first

In practice: for time series, very popular to use "dynamic time warping" to first align (it works kind of like how spell check does for words)

Is a Similarity Function Any Good?

Easy thing to check:

- Pick a data point
- Compute its similarity to all the other data points, and sort them from most similar to least similar
- Inspect the most similar data points

If the most similar points are not interpretable, it's quite likely that your similarity function isn't very good =(

Going from Similarities to Clusters

There’s a whole zoo of clustering methods

Two main categories we'll talk about (time permitting…):

Generative models

1. Pretend data generated by specific model with parameters

2. Learn the parameters ("fit model to data")

3. Use fitted model to determine cluster assignments

We start here

Hierarchical clustering

Top-down: Start with everything in 1 cluster and decide on how to recursively split

Bottom-up: Start with everything in its own cluster and decide on how to iteratively merge clusters

We're going to start with perhaps the most famous of **clustering methods**

It won't yet be apparent what this method has to do with generative models

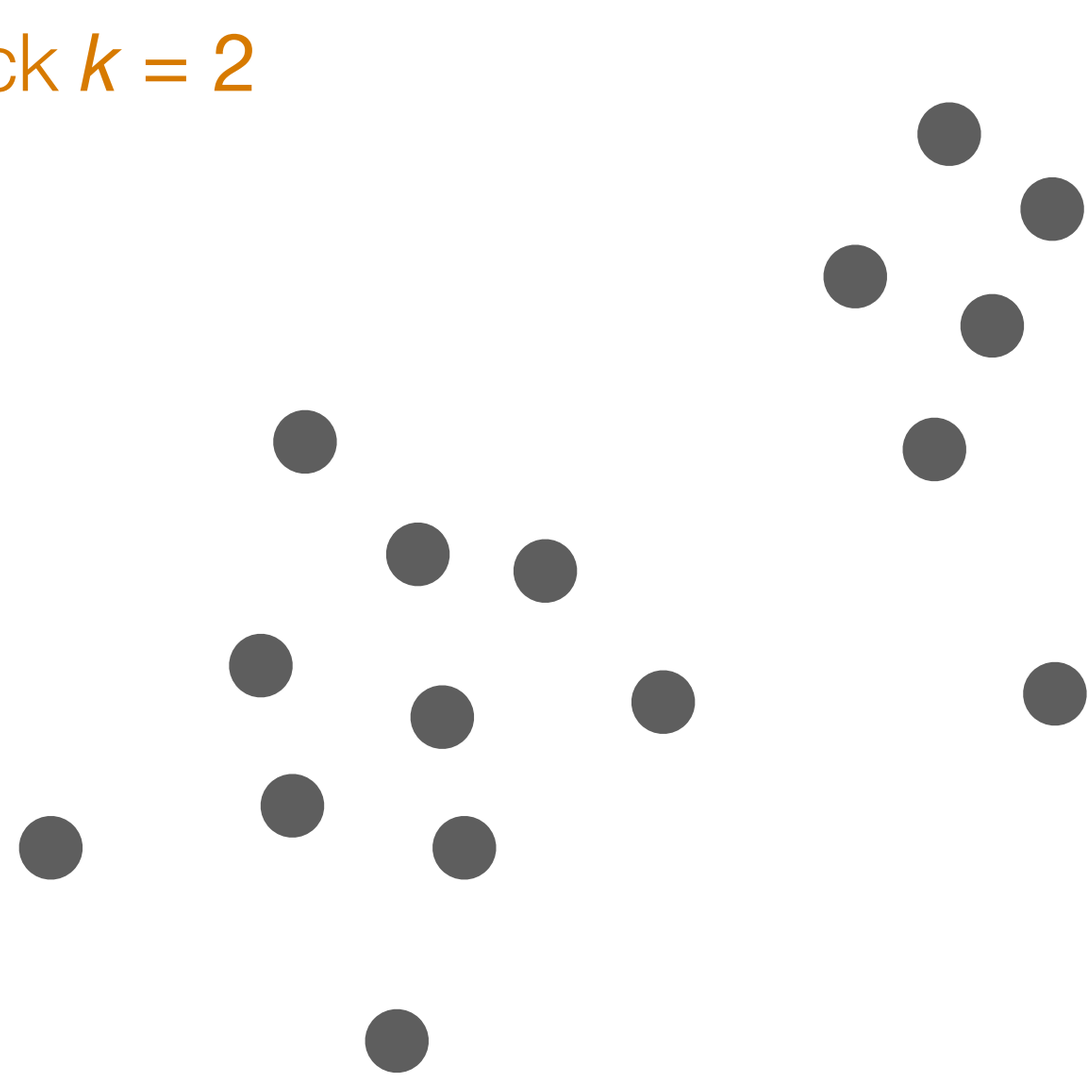
Similarity determined via Euclidean distance

k-means

Step 0: Pick *k*

We'll pick *k* = 2

Step 1: Pick guesses for where cluster centers are



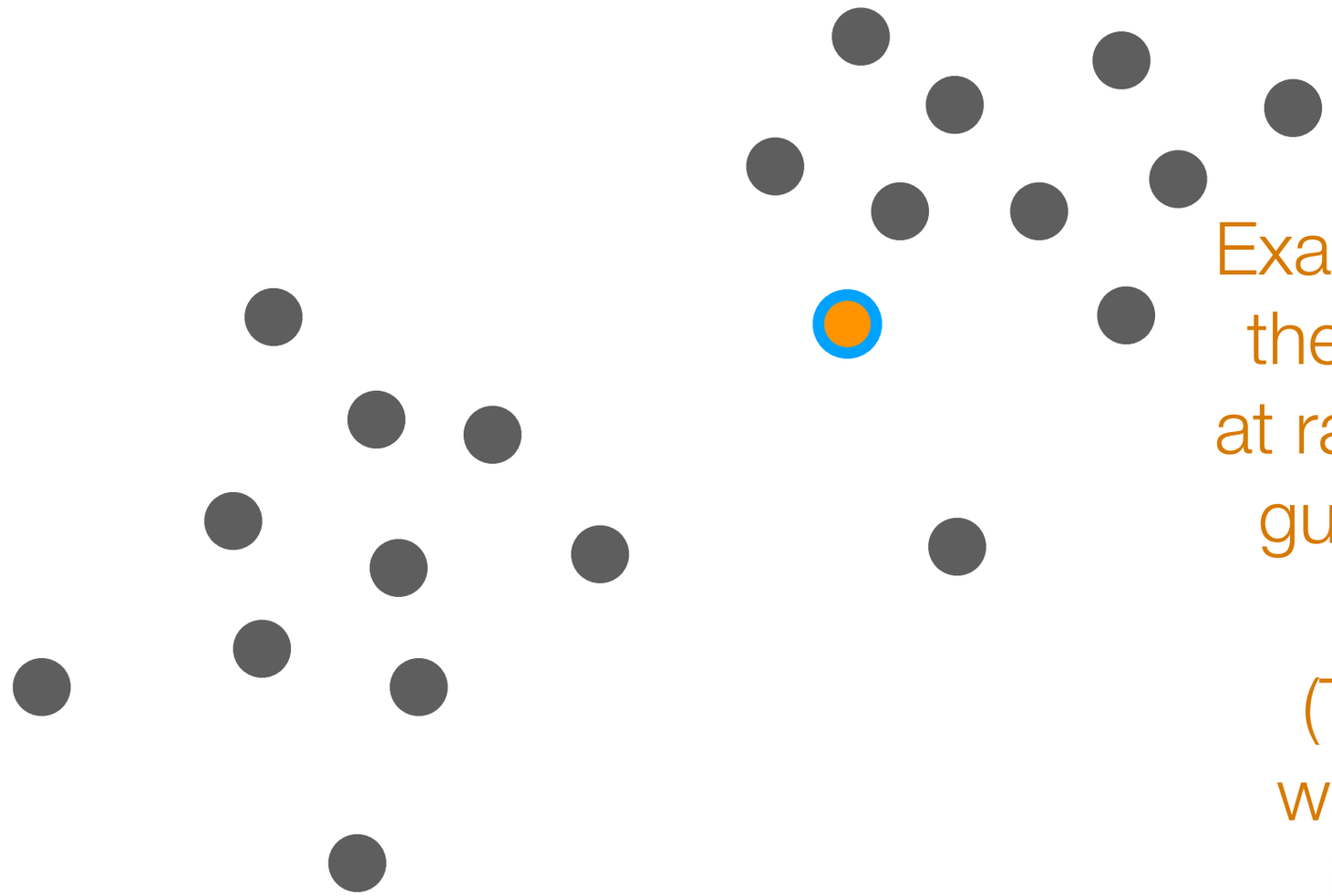
Example: choose *k* of the points uniformly at random to be initial guesses for cluster centers

k-means

Step 0: Pick *k*

We'll pick *k* = 2

Step 1: Pick guesses for where cluster centers are



Example: choose *k* of the points uniformly at random to be initial guesses for cluster centers

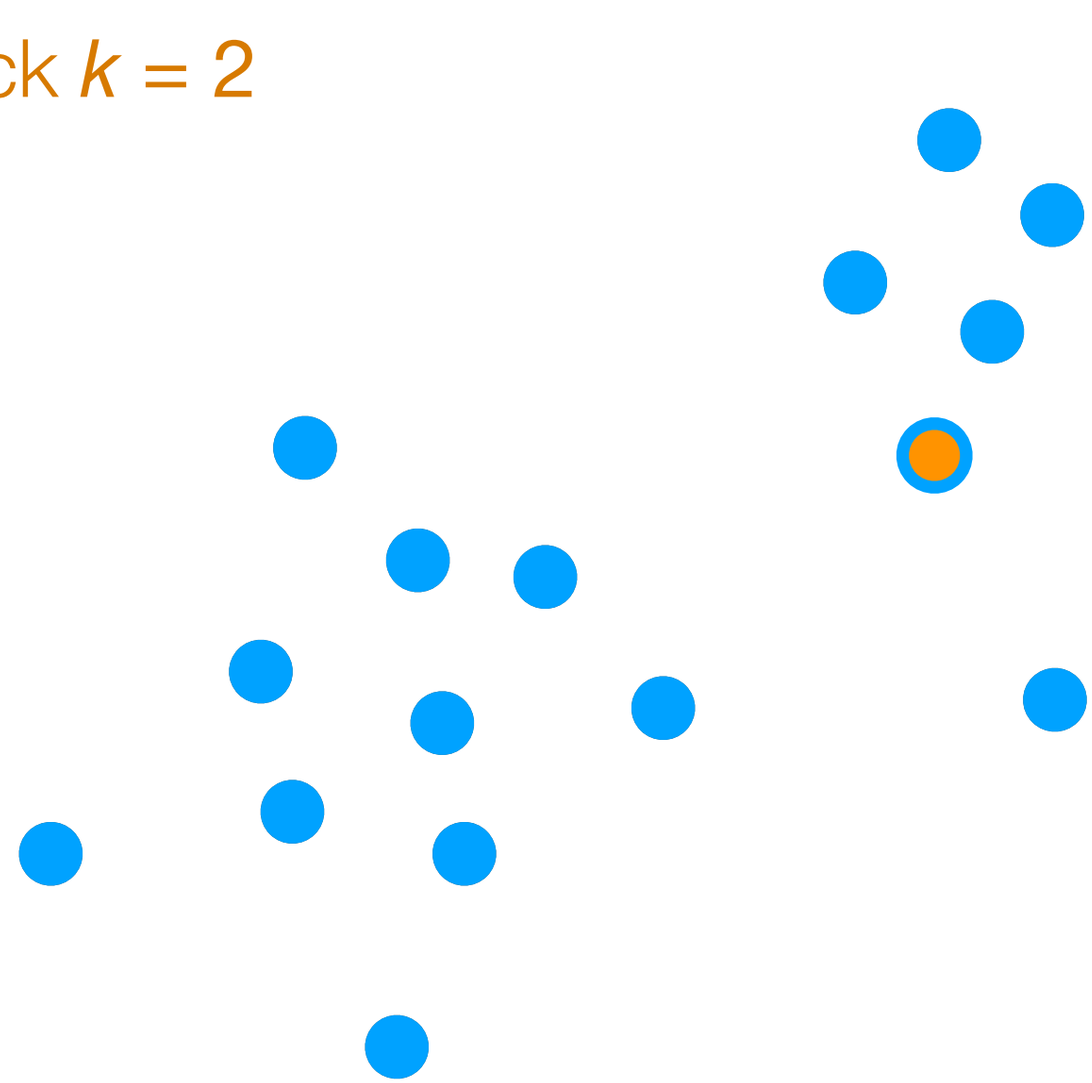
(There are many ways to make the initial guesses)

k-means

Step 0: Pick *k*

We'll pick *k* = 2

Step 1: Pick guesses for where cluster centers are



Example: choose *k* of the points uniformly at random to be initial guesses for cluster centers

(There are many ways to make the initial guesses)

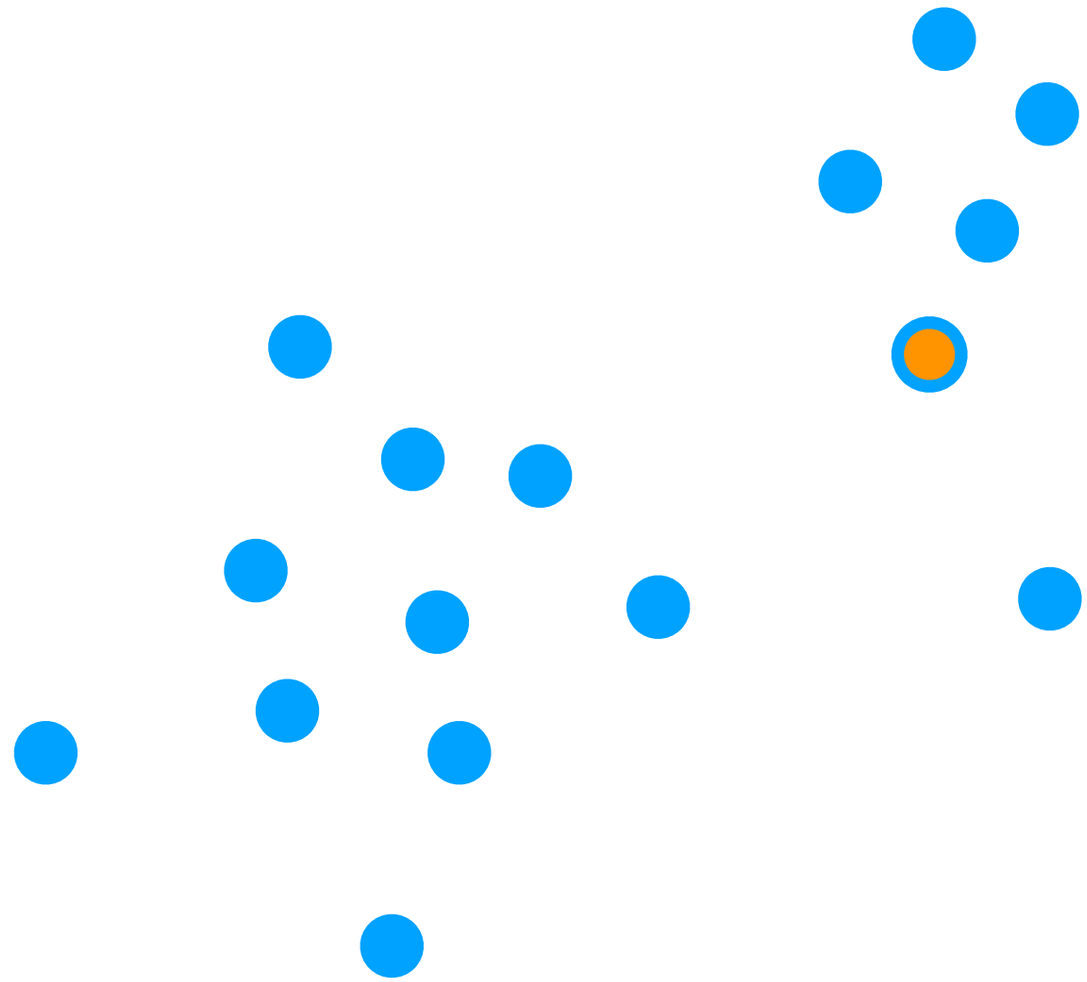
Step 2: Assign each point to belong to the closest cluster

k-means

Step 0: Pick *k*

We'll pick *k* = 2

Step 1: Pick guesses for where cluster centers are



Example: choose *k* of the points uniformly at random to be initial guesses for cluster centers

(There are many ways to make the initial guesses)

Step 2: Assign each point to belong to the closest cluster

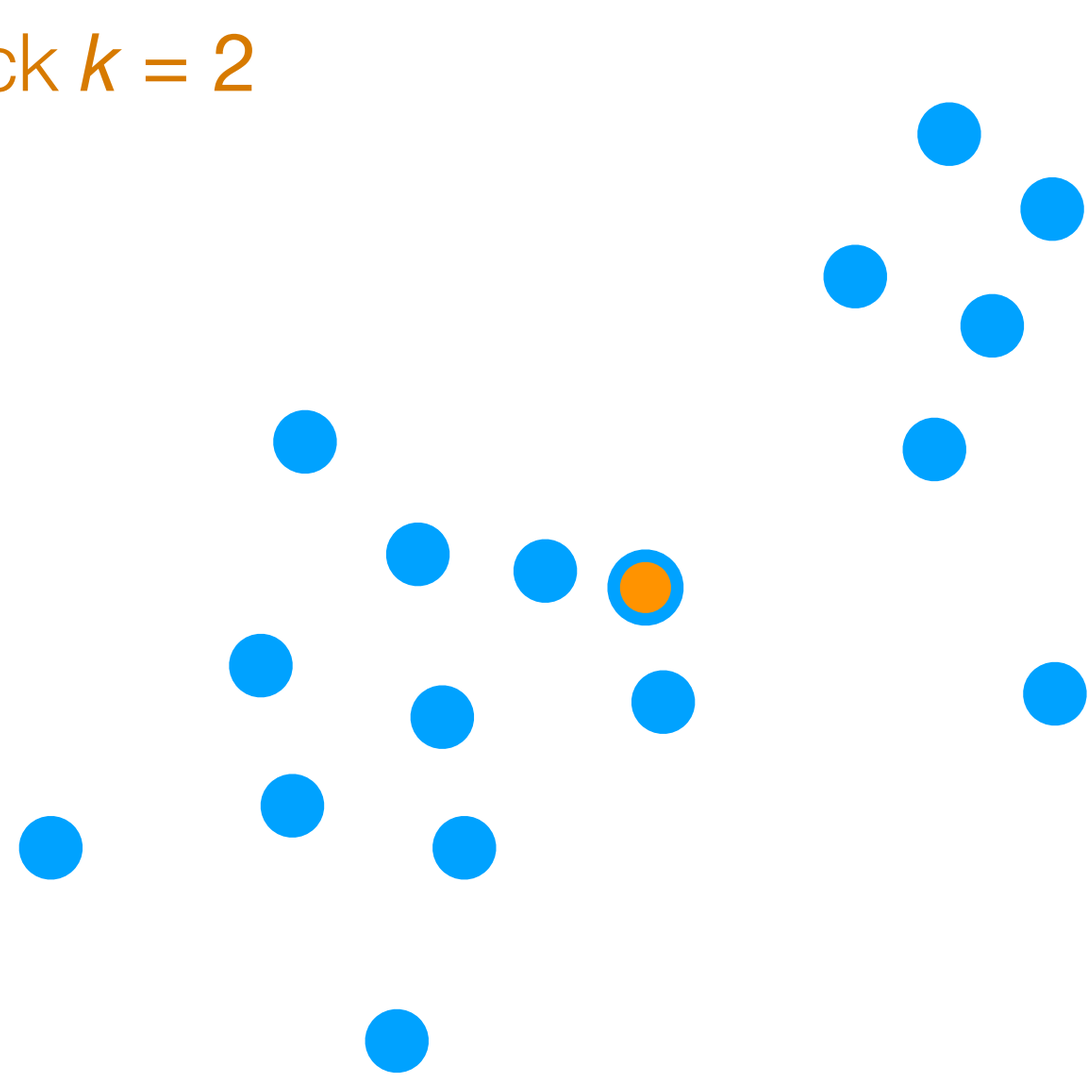
Step 3: Update cluster means (to be the center of mass per cluster)

k-means

Step 0: Pick *k*

We'll pick *k* = 2

Step 1: Pick guesses for where cluster centers are



Example: choose *k* of the points uniformly at random to be initial guesses for cluster centers

(There are many ways to make the initial guesses)

Step 2: Assign each point to belong to the closest cluster

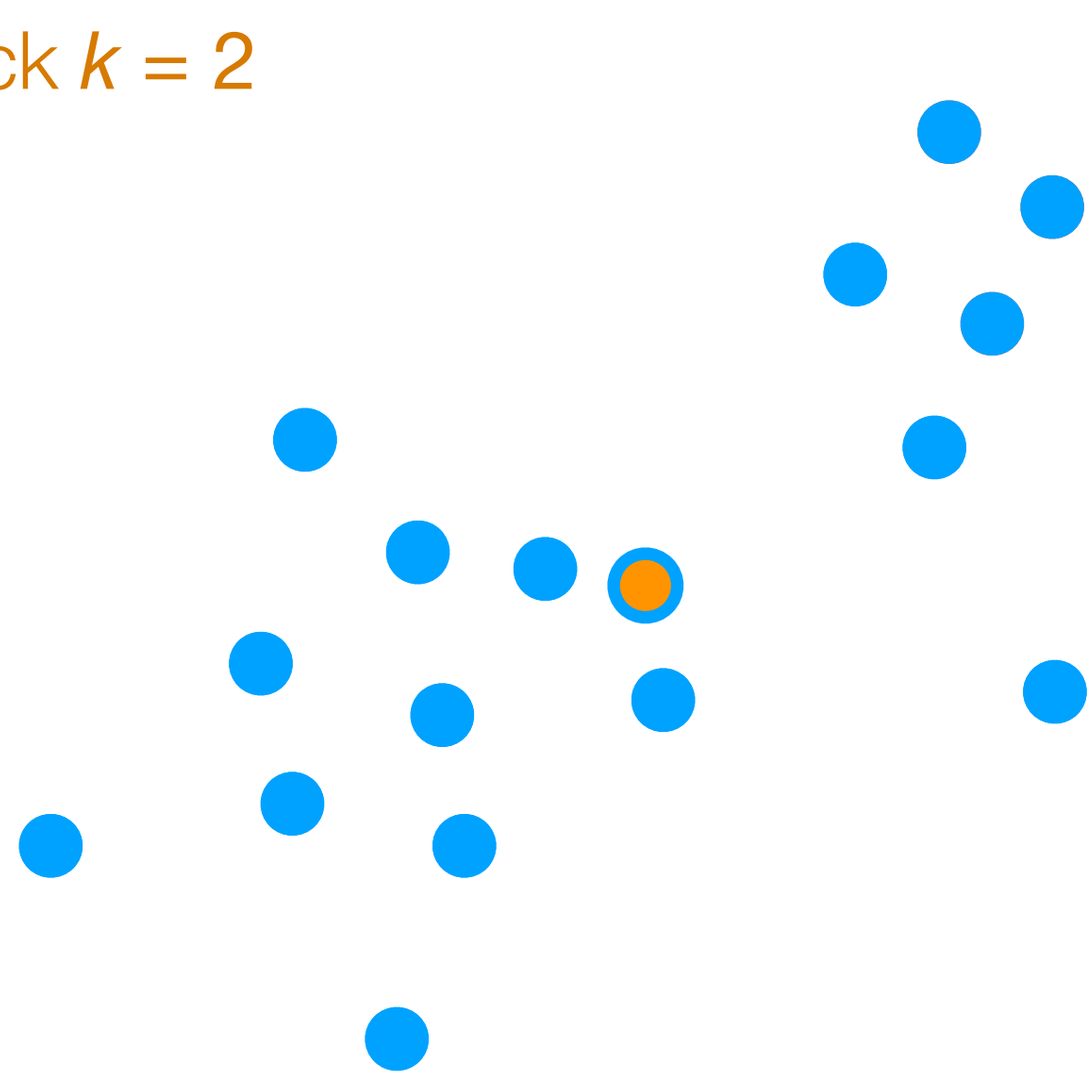
Step 3: Update cluster means (to be the center of mass per cluster)

k-means

Step 0: Pick *k*

We'll pick *k* = 2

Step 1: Pick guesses for where cluster centers are



Example: choose *k* of the points uniformly at random to be initial guesses for cluster centers

(There are many ways to make the initial guesses)

Repeat Step 2: Assign each point to belong to the closest cluster

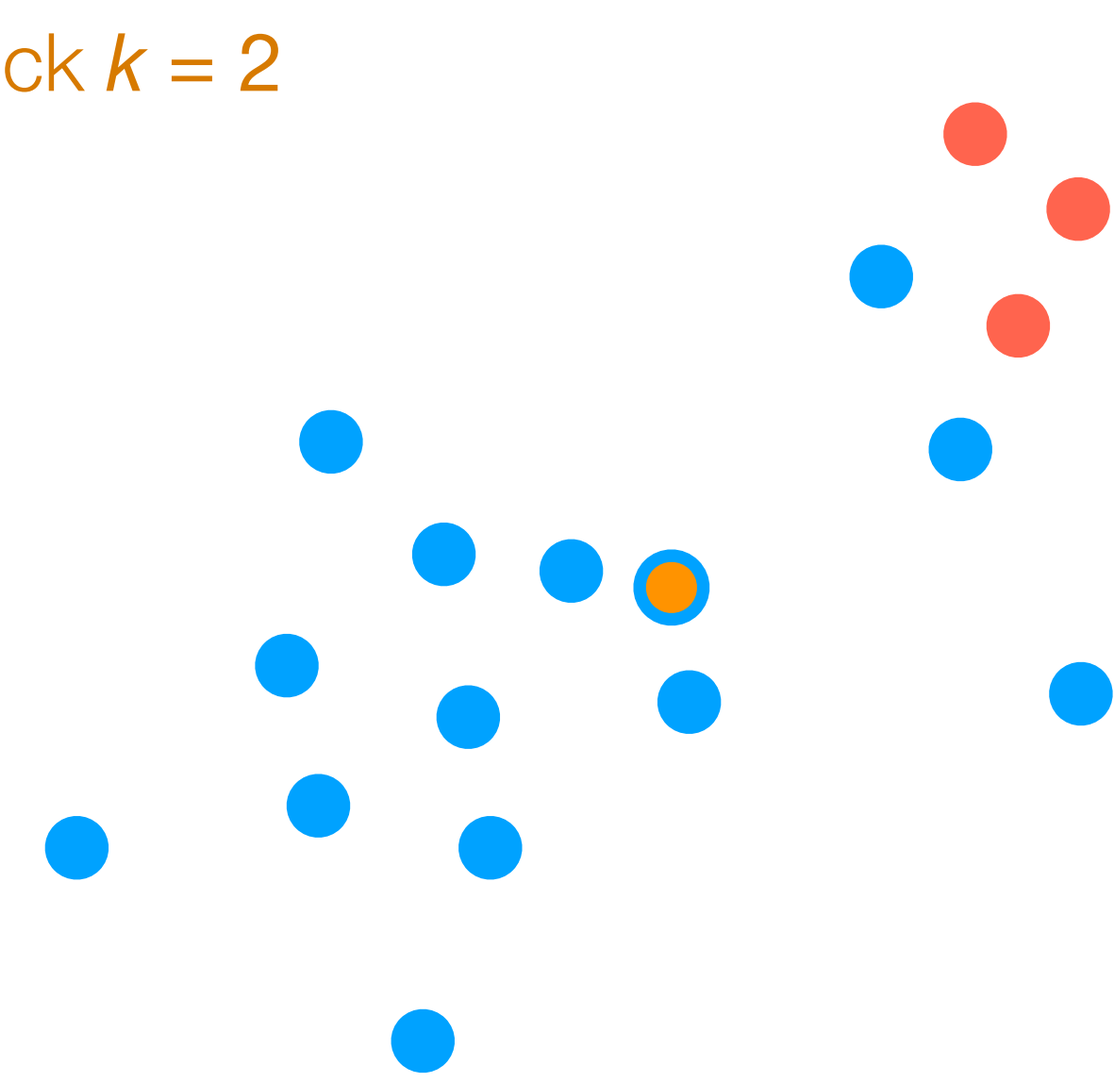
Step 3: Update cluster means (to be the center of mass per cluster)

k-means

Step 0: Pick *k*

We'll pick *k* = 2

Step 1: Pick guesses for where cluster centers are



Example: choose *k* of the points uniformly at random to be initial guesses for cluster centers

(There are many ways to make the initial guesses)

Step 2: Assign each point to belong to the closest cluster

Repeat

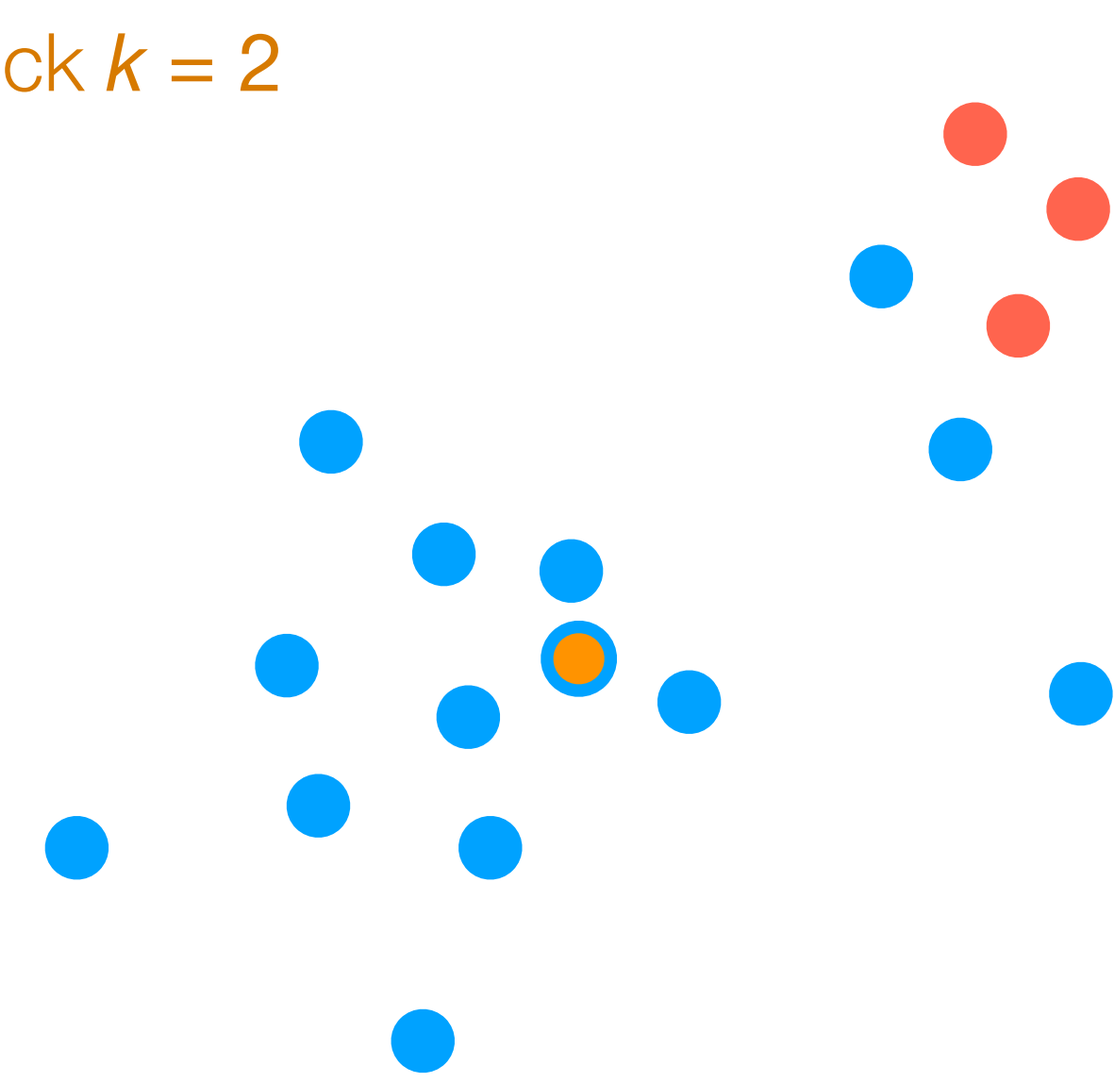
Step 3: Update cluster means (to be the center of mass per cluster)

k-means

Step 0: Pick *k*

We'll pick *k* = 2

Step 1: Pick guesses for where cluster centers are



Example: choose *k* of the points uniformly at random to be initial guesses for cluster centers

(There are many ways to make the initial guesses)

Step 2: Assign each point to belong to the closest cluster

Repeat

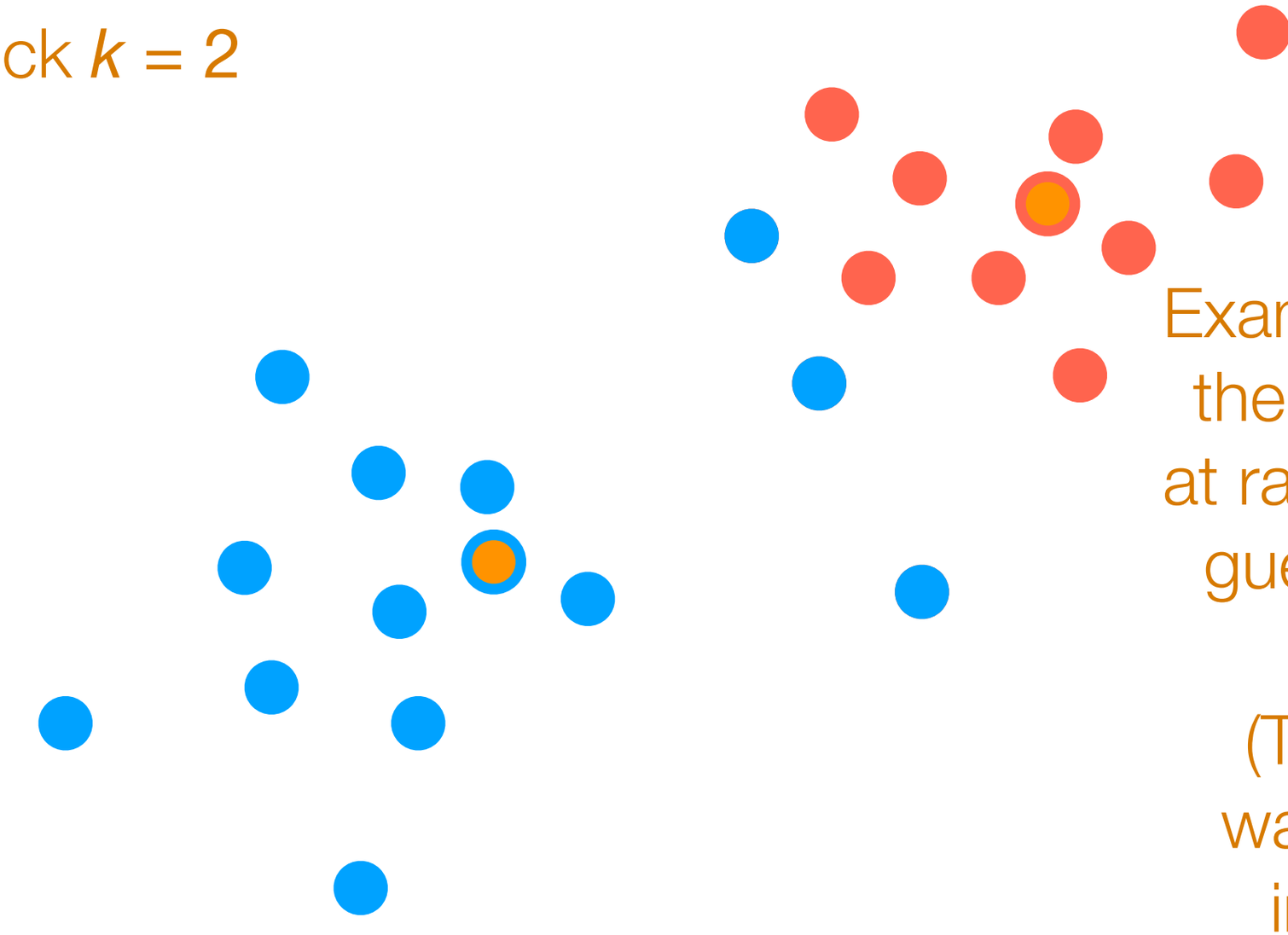
Step 3: Update cluster means (to be the center of mass per cluster)

k-means

Step 0: Pick *k*

We'll pick *k* = 2

Step 1: Pick guesses for where cluster centers are



Example: choose *k* of the points uniformly at random to be initial guesses for cluster centers

(There are many ways to make the initial guesses)

Repeat Step 2: Assign each point to belong to the closest cluster

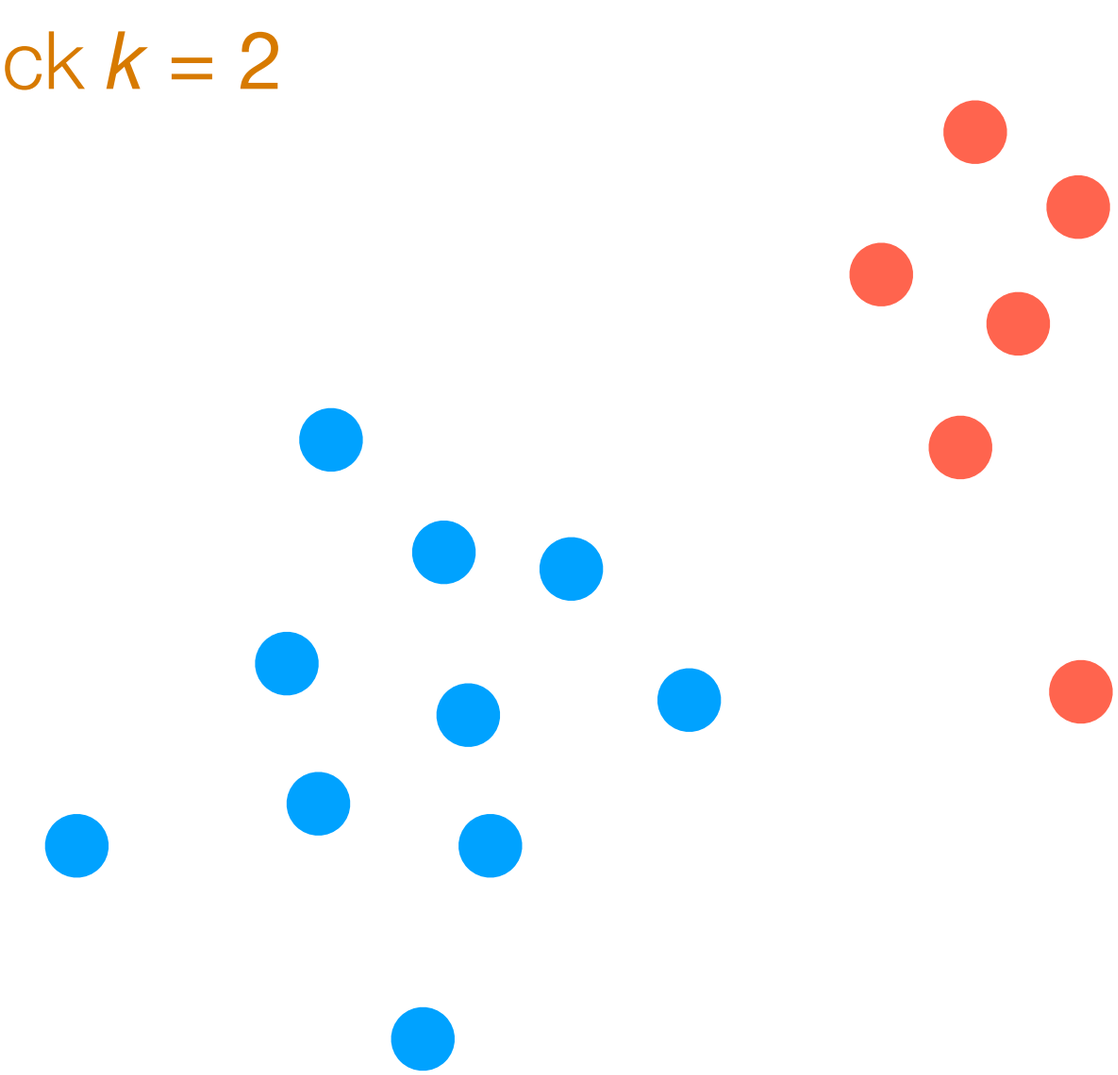
Step 3: Update cluster means (to be the center of mass per cluster)

k-means

Step 0: Pick *k*

We'll pick *k* = 2

Step 1: Pick guesses for where cluster centers are



Example: choose *k* of the points uniformly at random to be initial guesses for cluster centers

(There are many ways to make the initial guesses)

Step 2: Assign each point to belong to the closest cluster

Repeat

Step 3: Update cluster means (to be the center of mass per cluster)

k-means

Step 0: Pick *k*

We'll pick *k* = 2

Step 1: Pick guesses for where cluster centers are

Example: choose *k* of the points uniformly at random to be initial guesses for cluster centers

(There are many ways to make the initial guesses)

Repeat until convergence:

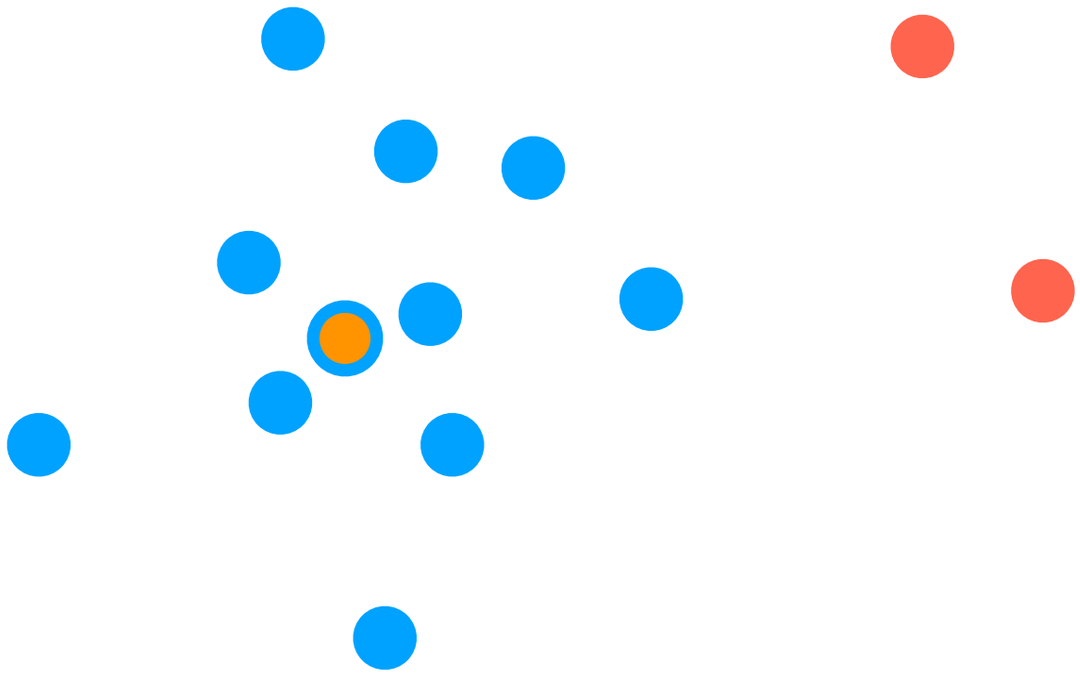
Step 2: Assign each point to belong to the closest cluster

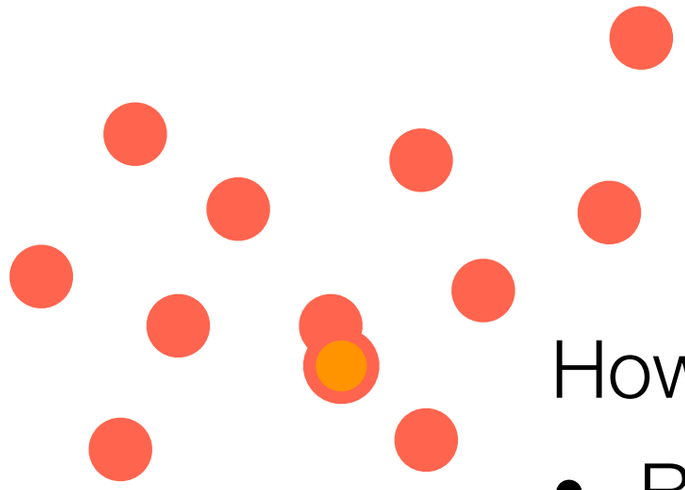
Step 3: Update cluster means (to be the center of mass per cluster)

k-means

Final output: cluster centers, cluster assignment for every point

Remark: Very sensitive to choice of *k* and initial cluster centers





How to pick *k*?

- Basic check: If you have really, really tiny clusters ⇒ decrease *k*
- More details later

Suggested way to pick initial cluster centers: "*k*-means++" method

(rough intuition: incrementally add centers; favor adding center far away from centers chosen so far)

k-means

Demo

When does *k*-means work well?

k-means is related to a more general model, which will help us understand *k*-means

Gaussian Mixture Model (GMM)

Demo